Zadaća 5.

Ova zadaća nosi ukupno 7 poena. Prvi i treći zadatak nose 1,3 poena, dok ostali zadaci nose 1,1 poen. Prva dva zadatka traže poznavanje prvih 12 predavanja i pretpostavljeno predznanje iz predmeta "Osnove računarstva", dok ostali zadaci traže također i poznavanje Predavanja 13 te Predavanja 14. Ova zadaća je od vitalnog značaja za razuijevanje kursa kao cjeline, a rok za njenu predaju je ponedjeljak, 22. VI 2020. do kraja dana.

1. U raznim oblastima matematike često se javlja potreba za radom sa matricama. Mada jezik C++ u svojim standardnim bibliotekama ne sadrži nikakve specijalističke tipove podataka za rad sa matricama, takvi tipovi se mogu naći u raznim nestandardnim bibliotekama koje se mogu posebno instalirati. Posebno se često javlja potreba za simetričnim matricama (npr. matrice koje se dobijaju metodom konturnih struja ili napona čvorova u električnim kolima bez zavisnih izvora su uvijek simetrične). Dobra je stvar što se simetrične matrice mogu realizirati tako da je utrošak memorije za smještanje njihovih elemenata dvostruko manji nego kod običnih matrica, a da pri tome korisnik koji radi sa takvom matricom ne primjećuje tu činjenicu. Vaš zadatak je upravo da kreirate generičku klasu "`SimetricnaMatrica`" koja će predstavljati jedan jednostavan tip podataka koji modelira simetrične matrice, pri čemu tip njihovih elemenata može biti proizvoljnog tipa. Sa ovakvim tipom podataka treba da budu moguće stvari poput sljedećih:

```
SimetricnaMatrica<double> a(5), b(5);
std::cout << "Unesi elemente matrice A: ";
std::cin >> a;
std::cout << "Unesi elemente matrice B: ";
std::cin >> b;
std::cout << "Matrica A+B glasi:\n";
std::cout << std::setw(10) << a + b;
```

   Jedini atribut klase treba da bude atribut koji predstavlja vektor vektôrā (odnosno koji je tipa "`vector`" čiji su elementi ponovo tipa "`vector`") u kojem se interno čuvaju elementi matrice, ali samo elementi koji su ispod glavne dijagonale ili na njoj (elementi iznad glavne dijagonale svakako su jednaki elementu ispod glavne dijagonale koji se nalazi simetrično u odnosu na dijagonalu). Drugim riječima, taj atribut će u suštini biti "grbava" matrica (organizirana kao vektor vektôrā) u kojoj prvi red ima jedan element, drugi red dva elementa, itd. Kako je ovo ujedno jedini atribut klase, tako da čitava klasa će zapravo predstavljati pogodan "omotač" oko klasične realizacije matrica kao vektora čiji su elementi ponovo vektori. Klasa treba da sadrži sljedeće elemente:

   - Konstruktor sa bez parametara, koji kreira praznu "matricu" formata 0×0 (njegova uloga je da dozvoli mogućnost kreiranja nizova čiji su elementi matrice).
   - Generički konstruktor koji prima kao parametar također objekat tipa "`SimetricnaMatrica`", ali sa drugačijim tipom elemenata. Ovaj konstrkutor bi trebao da omogući recimo da se simetrična matrica čiji su elementi tipa "**`double`**" može inicijalizirati simetričnom matricom čiji su elementi recimo tipa "**`int`**" (naravno, to će raditi samo ukoliko je tip elemenata izvorne matrice konvertibilan u tip elemenata odredišne matrice). Ovaj konstruktor će indirektno (putem automatske pretvorbe tipa) omogućiti i međusobno dodjeljivanje simetričnih matrica različitih tipova elemenata. Napomena: konstruktori i funkcije članice također mogu biti generički (to se koristi recimo ukoliko je tip parametara djelimično ili potpuno nepoznat), mada to nije eksplicitno naglašeno na predavanju. U tom slučaju, "**`template`**" deklaraciju treba staviti ispred deklaracije ili definicije konstruktora odnosno funkcije članice, kao i u slučaju običnih generičkih funkcija.
   - Konstruktor sa koji prima jedan parametar tipa vektora vektôrā, čiji su elementi istog tipa kao što je pretpostavljeni tip elemenata matrice. Ovaj konstruktor kreira simetričnu matricu na osnovu elemenata vektora vektôrā koji mu je proslijeđen kao parametar. Osnovna namjena ovog konstruktora je da podrži automatsku pretvorbu vektora vektôrā u odgovarajući tip "`SimetricnaMatrica`". Zadani vektor vektôrā mora imati prikladnu formu (prvi red jedan element, drugi red dva elementa, itd.), u suprotnom treba baciti izuzetak tipa "`logic_error`" uz prateći tekst "Nekorektna forma simetricne matrice". Napomena: zbog postojanja automatske konverzije inicijalizacionih listi u vektore, ovaj konstruktor će automatski podržati i konstrukcije poput "`SimetricnaMatrica<double> m({{1}, {2, 3}, {4, 5, 6})`".

- Sekvencijski konstruktor za kreiranje simetričnih matrica direktno iz inicijalizacijskih listi, tj. za podršku konstrukcija poput "`SimetricnaMatrica<double> m{{1}, {2, 3}, {4, 5, 6}}`" (primijetite da u ovom slučaju, za razliku od prethodnog konstruktora, ovdje nema dodatnih okruglih zagrada). Slično prethodnom konstruktoru, i ovaj konstruktor baca izuzetak ukoliko nije zadana ispravna forma matrice.
- Konstruktor sa jednim cjelobrojnim parametrom n, koji kreira simetričnu matricu formata $n \times n$ (simetrične matrice uvijek imaju isti broj redova i kolona) čiji su svi elementi nule. Ukoliko je $n < 0$, baca se izuzetak tipa "`domain_error`" uz prateći tekst "Neispravna dimenzija".
- Metodu "`DajDimenziju`" bez parametara, koja vraća kao rezultat broj redova odnosno broj kolona matrice (ovo dvoje je uvijek isto za simetrične matrice).
- Preklopljen unarni operator "`!`", koji primijenjen na matricu daje "**`true`**" ukoliko je matrica nul-matrica (tj. ukoliko su joj svi elementi nule), a "**`false`**" u suprotnom.
- Preklopljene binarne operatore "`+`", "`-`" i "`*`" koji redom nalaze zbir, razliku i proizvod simetričnih matrica, pod uvjetom da su matrice međusobno saglasne za izvođenje tih operacija. U suprotnom, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Matrice nisu saglasne za trazenu operaciju". Zbir i razlika simetričnih matrica su i sami simetrične matrice, ali produkt simetričnih matrica ne mora biti simetrična matrica. Stoga, ukoliko bi množenje dalo matricu koja nije simetrična, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Rezultat nije simetricna matrica". Matrice ne moraju biti istog tipa elemenata (recimo, može se sabrati matrica čiji su elementi tipa "**`int`**" sa matricom čiji su elementi tipa "**`double`**", pri čemu rezultirajuća matrica treba imati onakav tip elemenata kakav se dobije primjenom odgovarajuće operacije nad elementima jedne i druge matrice (uputa: ovdje će Vam trebati "**`decltype`**" operator). Operator "`*`" također treba da podržava množenje broja sa matricom odnosno matrice sa brojem. Tip broja ne mora biti isti kao tip elemenata matrice. Odgovarajuće operatorske funkcije obavezno treba implementirati *izvan deklaracije klase* (upadnete li u probleme, a skoro je sigurno da hoćete, informacije date pred kraj Predavanja 11_b mogu Vam biti od velike koristi).
- Prekopljene binarne operatore "`+=`","`*=`" i "`-=`" takve da izrazi poput "`X += Y`", "`X -= Y`" i "`X *= Y`" uvijek imaju isti efekat kao i izrazi "`X = X + Y`", "`X = X - Y`" i "`X = X * Y`" kad god to ima smisla. Pri tome, realizacije ovih operatora se ne smiju trivijalno svoditi na realizacije operatora "`+`","`-`" i "`*`" (tj. nije dozvoljena izvedba operatora "`+=`" koja se prosto sastoji od naredbe poput "**`return`** `x = x + y`"). S druge strane, dozvoljeno je obrnuto (čak je i poželjno), odnosno realizirati operator "`+`" preko "`+=`" itd. tamo gdje je to moguće.
- Preklopljene binarne operatore "`==`" i "`!=`" koji daju rezultat "**`true`**" ukoliko su operandi jednake odnosno različite matrice, i "**`false`**" u suprotnom. Matrice se smatraju jednakim ukoliko im je ista dimenzija i ukoliko im elementi na istim pozicijama imaju iste vrijednosti.
- Preklopljen operator "`()`" koji omogućava pristup elementima simetrične matrice zadavanjem reda i kolone kao parametara u zagradi, npr. "`m(2, 3)`" je element u drugom redu i trećoj koloni. Indeksi se gledaju "matematički", dakle od jedinice a ne od nule. Vodite računa da moraju biti legalni svi indeksi u opsegu od jedinice do dimenzije matrice, bez obzira što elementi iznad glavne dijagonale nisu fizički pohranjeni u matrici (pokušaj pristupa takvom elementu treba preusmjeriti na njegovog "blizanca" sa druge strane dijagonale). Ukolilo indeksi nisu u dozvoljenom opsegu, treba baciti izuzetak tipa "`range_error`" uz prateći tekst "Nelegalni indeksi". Rezultat treba biti referenca na element, tako da se može koristiti i sa lijeve strane operatora dodjele, osim u slučaju kada se ovaj operator primijeni na konstantnu matricu (tada umjesto reference treba vratiti kopiju elementa).
- Preklopljen operator "`[]`" koji omogućava pristup elementima simetrične matrice sintaksom kao da se radi o klasičnim dvodimenzionalnim nizovima (npr. "`m[5][2]`", sa indeksacijom *od nule* i *bez provjere* ispravnosti indeksa. I ovdje treba biti podržan pristup svim elementima, čak i onima iznad glavne dijagonale koji nisu fizički pohranjeni u matrici (u tom slučaju se također vrši preusmjeravanje na odgovarajući element ispod glavne dijagonale). Potrebno je podržati konzistentan tretman konstantnih matrica (tj. da se za slučaj konstantnih matrica rezultat ne može koristiti sa lijeve strane operatora dodjele). Uputa: ovo nije tako lako izvesti, s obzirom da se izraz poput "`m[i][j]`" tretira kao "(`m.`**`operator`**`[](i))[j]`", a problem je što nam treba i specijalan tretman drugog indeksa (radi već spomenutog preusmjeravanja). Stoga operator "`[]`" primijenjen na matricu "`m`" (tj. izraz "`m[i]`") treba da vrati neki pomoćni objekat (nekog pomoćnog tipa) koji u sebi čuva pokazivač ili referencu na odgovarajući red matrice, a koji također ima preklopljen operator "`[]`", koji će konačno vratiti referencu na odgovarajući element (ili njegovu kopiju za slučaj konstantne matrice) kojem se pristupa.

- Preklopljen operator "`<<`" koji treba da podrži ispis simetrične matrice na izlazni tok. Elementi matrice se ispisuju red po red (elementi jednog reda matrice u jednom redu, a nakon svakog reda matrice prelazi se u novi red). Ispisuje se *kompletna matrica*, bez obzira što je zapravo samo oko polovine njenih elemenata efektivno pohranjeno. Svaki element treba zauzeti onoliko prostora koliko je postavljeno manipulatorom "`setw`" ili pozivom funkcije "`width`" nad objektom toka. Uputa: da biste unutar operatorske funkcije saznali kolika je trenutna postavka širine, pozovite funkciju "`width`" nad objektom toka, ali *bez parametara*. Ono što dobijete kao rezultat je trenutna postavka širine.
- Preklopljen operator "`>>`" koji treba da podrži unos simetrične matrice sa ulaznog toka. Matrica se unosi prosto kao slijed elemenata matrice (jedan za prvi red, dva za drugi, tri za treći, itd.) koji su razmaknuti bjelinom (razmakom, tabulatorom ili prelaskom u novi reda).

Sve metode implementirajte izvan deklaracije klase, osim trivijalnih metoda koje se mogu izvesti u najviše dvije naredbe. Metode koje su inspektori treba deklarirati kao takve. Obavezno napišite i testni program u kojem ćete demonstrirati sve elemente napisane generičke klase, na bar dva različita tipa elemenata.

2. Za potrebe neke meteorološke stanice neophodno je vrsiti čestu registraciju količine padavina. Za tu svrhu meteorološka stanica koristi računarski program u kojem je definirana i implementirana klasa nazvana "`Padavine`". Ova klasa omogućava čuvanje podataka o količini padavina za izvjesni vremenski period u vektoru cijelih brojeva, kojem se pristupa preko odgovarajućeg privatnog atributa (količina padavina se zadaje u centimetrima, zaokruženo na cijeli broj). Interfejs klase sadrži sljedeće elemente:

   - Konstruktor sa jednim parametrom koji predstavlja maksimalno dozvoljenu količinu padavina koja se može registrirati (oprez: ovo nije maksimalan broj količina padavina koje se mogu registrirati, nego maksimalan iznos količine padavina koji se smije zadati pri jednoj registraciji). Ovaj parametar mora biti pozitivan, u suprotnom treba baciti izuzetak tipa "`range_error`" uz odgovarajući prateći tekst "Ilegalna maksimalna kolicina". Potrebno je zabraniti da se ovaj konstruktor koristi za automatsku konverziju cijelih brojeva u objekte tipa "`Padavine`", s obzirom da bi takva konverzija bila besmislena.
   - Metodu "`RegistrirajPadavine`" koja vrši registraciju nove količine padavina, pri čemu se količina padavina koja se registrira prenosi kao parametar metode. U slučaju je količina padavina manja od nule ili veća od maksimalne dozvoljene količine padavina, metoda treba da baci izuzetak tipa "`range_error`" uz prateći tekst "Ilegalna kolicina padavina".
   - Metodu "`DajBrojRegistriranihPadavina`" koja daje broj registriranih količina padavina.
   - Metodu "`BrisiSve`" koja briše sve unesene količine padavina.
   - Metode "`DajMinimalnuKolicinuPadavina`" i "`DajMaksimalnuKolicinuPadavina`" koje vraćaju kao rezultat minimalnu i maksimalnu količinu padavina (u slučaju da nema registriranih količina padavina, obje metode treba da bace izuzetak tipa "`range_error`" uz prateći tekst "Nema registriranih padavina"). Za realizaciju nije dozvoljeno koristiti petlje, nego isključivo odgovarajuće funkcije iz biblioteke "`algorithm`".
   - Metodu "`DajBrojDanaSaPadavinamaVecimOd`" koja vraća kao rezultat broj dana u kojima je količina padavina bila veća od vrijednosti koja se zadaje kao parametar (u slučaju da nema registriranih količina padavina, metode treba da bace izuzetak tipa "`range_error`" uz prateći tekst "Nema registriranih padavina"). Za realizaciju nije dozvoljeno koristiti petlje, niti lambda funkcije ili pomoćne imenovane funkcije kriterija, nego isključivo samo odgovarajuće funkcije i/ili funktore iz biblioteka "`algorithm`" i "`functional`". Napomena: Ovdje ćete morati koristiti veznike, a na njihovu upotrebu treba se navići. Prvo probajte riješiti problem uz pomoć lambda funkcija. Kada Vam funkcija proradi, probajte istu funkcionalnost postići pomoću jednostavnijih ali zastarjelih veznika "`bind1st`" ili "`bind2st`". (koji su, usput, u verziji C++17 potpuno odbaćeni), kao međukorak da bolje steknete osjećaj šta se zaista dešava. Kada Vam i to proradi, zamijenite zastarjele veznike sa boljim i univerzalnijim veznikom "`bind`" (zastarjele verzije veznika nemojte ostavljati u završnoj verziji).
   - Metodu "`Ispisi`" koja ispisuje sve unesene (registrirane) količine padavina sortirane u opadajućem poretku (tj. najveća količina padavina se ispisuje prva), pri čemu se svaka količina padavina ispisuje u posebnom redu. Pri tome je neophodno koristiti funkcije i/ili funktore iz biblioteka "`algorithm`" i "`functional`" (upotreba lambda funkcija ili pomoćnih imenovanih funkcija nije dozvoljena). Ova funkcija obavezno treba biti inspektor funkcija, tj. treba biti deklarirana sa modifikatorom "**`const`**"!

- Preklopljeni operator "`[]`" koji omogućava da se direktno pročita *i*-ta registrirana količina padavina (numeracija ide od jedinice). Ukoliko je indeks izvan dozvoljenog opsega, treba baciti izuzetak tipa "`range_error`" uz prteći tekst "Neispravan indeks". Pri tome, ovaj operator se *ne može koristiti* za izmjenu podataka, odnosno ne može se koristiti sa lijeve strane znaka jednakosti.
- Preklopljeni operator "`++`" koji povećava sve registrirane količine padavina za jedinicu (pri tome se za jedinicu povećava i informacija o maksimalno dozvoljenoj količini padavina). Potrebno je podržati kako prefiksnu, tako i postfiksnu verziju ovog operatora. Za realizaciju nije dozvoljeno koristiti petlje, niti lambda funkcije ili pomoćne imenovane funkcije, nego isključivo odgovarajuće funkcije i/ili funktore iz biblioteka "`algorithm`" i "`functional`". Vrijedi ista napomena kao u metodi "`DajBrojDanaSaPadavinamaVecimOd`".
- Preklopljene operatore "`+`" i "`-`" koji djeluju na sljedeći način: Ukoliko je "`X`" objekat tipa "`Padavine`", a "`Y`" cijeli broj, tada je "`X + Y`" novi objekat tipa "`Padavine`" u kojem su sve registrirane količine padavina povećane za iznos "`Y`" (u novodobijenom objektu treba povećati i informaciju o maksimalno dozvoljenoj količini padavina). Izraz "`Y + X`" treba da ima isto značenje kao i izraz "`X + Y`". Izraz "`X - Y`" u slučaju da je "`X`" objekat tipa "`Padavine`", a "`Y`" cijeli broj interpretira se analogno (uz odgovarajuće smanjenje informacije o maksimalno dozvoljenoj količini padavina), dok je tada "`Y - X`" objekat tipa "`Padavine`" u kojem su sve registrirane količine padavina oduzete od vrijednosti "`Y`", dok je maksimalna dozvoljena količina padavina upravo "`Y`". Pri tome, ukoliko se kao rezultat sabiranja ili oduzimanja dobije da neka od količina padavina postane negativna, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Nekorektan rezultat operacije". U slučaju kada su i "`X`" i "`Y`" objekti tipa "`Padavine`", tada je izraz "`X - Y`" novi objekat tipa "`Padavine`" koji sadrži *razlike* odgovarajućih količina padavina iz objekata "`X`" i "`Y`". U ovom posljednjem slučaju se podrazumijeva da "`X`" i "`Y`" sadrže isti broj registriranih količina padavina, kao i da su registrirane padavine u objektu "`X`" veće ili jednake od odgovarajućih registriranih padavina u objektu "`Y`" (u suprotnom, treba baciti izuzetak tipa "`domain_error`" uz prateći tekst "Nesaglasni operandi"). Maksimalna dozvoljena količina padavina u novokreiranom objektu treba biti kakva je bila u objektu "`X`". U svim ostalim slučajevima, značenje izraza "`X + Y`" odnosno "`X - Y`" nije definirano. Za realizaciju ovih operatora nije dozvoljeno koristiti petlje, niti lambda funkcije ili pomoćne imenovane funkcije, nego isključivo odgovarajuće funkcije i/ili funktore iz biblioteka "`algorithm`" i "`functional`". Vrijedi ista napomena kao u metodi "`DajBrojDanaSaPadavinamaVecimOd`".
- Preklopljene operatore "`+=`" i "`-=`" čiji je cilj da značenje izraza oblika "`X += Y`" odnosno "`X -= Y`" bude identično značenju izraza "`X = X + Y`" i "`X = X - Y`" kad god oni imaju smisla.
- Preklopljeni unarni operator "`-`" koji daje kao rezultat novi objekat tipa "`Padavine`" u kojem su sve količine padavina oduzete od maksimalno dozvoljene količine padavina. Za realizaciju nije dozvoljeno koristiti petlje, niti lambda funkcije ili pomoćne imenovane funkcije, nego isključivo odgovarajuće funkcije i/ili funktore iz biblioteka "`algorithm`" i "`functional`". Vrijedi ista napomena kao u metodi "`DajBrojDanaSaPadavinamaVecimOd`".
- Preklopljene relacione operatore "`==`" i "`!=`" koje ispituju da li su dva objekta tipa "`Padavine`" jednaka ili nisu. Dva objekta ovog tipa smatraju se jednakim ukoliko sadrže isti broj registriranih količina padavina, i ukoliko su sve odgovarajuće registrirane količine padavina oba objekta jednake. Za realizaciju ovih operatora nije dozvoljeno koristiti petlje, nego isključivo odgovarajuće funkcije i/ili funktore iz biblioteka "`algorithm`" i "`functional`".

Implementirajte klasu sa navedenim svojstvima. Sve neophodne atribute treba obavezno izvesti kao privatne članove klase, a sve metode implementirajte izvan klase, osim metoda čija je implementacija dovoljno kratka, u smislu da zahtijeva recimo jednu ili dvije naredbe. Sve metode koje su po prirodi inspektori obavezno treba deklarirati kao takve. Obavezno napišite i mali testni program u kojem će se testirati sve elemente napisane klase.

3. Uprava ETF-a odlučila je da uvede automatizirani fast-food restoran. Za potrebe automatske obrade narudžbi, potrebno je razviti program zasnovan na skupini klasa, prema opisu koji slijedi.

   Klasa "`Narudzba`" opisuje najjednostavniju narudžbu, koja se sastoji samo od obroka. Ova klasa posjeduje konstruktor sa tri parmetra koji redom predstavljaju naziv obroka (tipa konstantni niz znakova, recimo "Burek"), cijenu obroka (realan broj) i puno ime (sa prezimenom) studenta koji je naručio obrok (također tipa konstantni niz znakova). Pored konstruktora, klasa sadrži trivijalne pristupne metode "`DajNazivObroka`", "`DajCijenuObroka`" i "`DajNarucioca`" pomoću

kojih se mogu saznati odgovarajuće informacije (prva i treća od ovih metoda daju kao rezultat objekat tipa "string", zatim metodu "Ispisi" koja ispisuje podatke o izvršenoj narudžbi (format ispisa biće preciziran nešto kasnije), kao i metodu "DajUkupnuCijenu" koja vraća ukupnu cijenu narudžbe. U slučaju klase "Narudzba", ova metoda će samo vratiti cijenu obroka pohranjenu u atributu, ali treba predvidjeti da će ova metoda biti eventualno izmijenjena u složenijim klasama naslijeđenim iz klase "Narudzba" kod kojih se ukupna cijena obroka formira na složeniji način. Konačno, klasa sadrži i metodu "DajKopiju" koja kreira dinamički alociranu kopiju objekta nad kojim je pozvana (ova metoda će se koristiti za polimorfno kopiranje). Bitno je da treba biti omogućeno da se preko pokazivača na klasu "Narudzba" uvijek pozivaju ispravne verzije svih metoda koje eventualno mogu biti promijenjene u naslijeđenim klasama, bez obzira na tip objekta na koji taj pokazivač pokazuje u trenutku poziva. Ispis koji vrši metoda "Ispisi" treba da izgleda poput sljedećeg:

```
Obrok: Burek
Cijena: 2.5 KM
Narucilac: Bil Klinton
```

Klasa "NarudzbaSaPicem" opisuje narudžbu koja uz obrok uključuje i piće. Ova klasa naslijeđena je iz klase "Narudzba". Konstruktor ove klase je dopunjen sa dodatna dva parametra koji predstvljaju naziv pića (npr. "Fanta", također po tipu konstantni niz znakova), te cijenu pića (realan broj). U skladu s tim, klasa ima i dvije dodatne pristupne metode "DajNazivPica" i "DajCijenuPica" koje vraćaju vrijednosti novouvedenih informacija (naziv pića se vraća kao objekat tipa "string"). Metoda koja vraća ukupnu cijenu narudžbe u ovoj klasi modificirana je tako da vrati ukupnu cijenu koja uključuje kako cijenu obroka, tako i cijenu pića. Isto tako, modificirane su i metoda za ispis podataka o narudžbi, koja sada treba uključiti i dopunske informacije koje nisu postojale u klasi "Narudzba", uključujući i informacije o ukupnoj cijeni, te metoda koja stvara dinamički kreiranu kopiju objekta. Konkretno, ispis koji proizvodi metoda "Ispisi" treba da izgleda poput sljedećeg:

```
Obrok: Sogan dolma
Pice: Fanta
Cijena: 4 KM
Cijena pica: 2.5 KM
Ukupna cijena: 6.5 KM
Narucilac: Josip Broz Tito
```

Klasa "Narudzbe" predstavlja kolekciju narudžbi, izvedenu kao vektor pokazivača na objekte tipa "Narudzba" ili "NarudzbaSaPicem". Klasa sadrži odgovarajući destruktor, koji oslobađa memoriju koju je objekat tipa "Narudzbe" zauzeo tokom svog života, te konstruktor kopije i preklopljeni operator dodjele koji omogućavaju bezbjedno kopiranje i međusobno dodjeljivanje objekata tipa "Narudzbe" zasnovano na strategiji dubokog kopiranja, uključujući i njihove optimizirane verzije koje se koriste za slučaj kada treba kopirati privremene objekte. Metode "NaruciObrok" i "NaruciObrokSaPicem" vrše kreiranje i dodavanje nove narudžbe, pri čemu se prva metoda odnosi na prostu narudžbu (obrok bez pića), dok se druga metoda odnosi na narudžbu obroka sa pićem. Parametri ovih metoda su isti kao i parametri konstruktora klasa "Narudzba" odnosno "NarudzbaSaPicem". One kreiraju novu narudžbu (u skladu sa navedenim parametrima) i smještaju ga u kolekciju. Metoda "ObradiNaruzdbu" nema parametara, a vrši ispis podataka o prvoj primljenoj narudžbi na ekran (tj. narudžbi koja najduže čeka na isporuku), pozivom odgovarajuće metode "Ispisi". Pored toga, ova metoda vrši brisanje obrađene narudžbe iz kolekcije, tako da će sljedeći poziv ove metode ispisati sljedeću naredbu po redu, i tako dalje, sve dok se ne obrade sve narudžbe. U slučaju da se ova metoda pozove kada je kolekcija prazna, metoda treba da baci izuzetak tipa "range_error" uz prateći tekst "Nema vise narudzbi". Metoda "DaLiImaNarudzbi" također nema parametar, a daje kao rezultat logičku vrijednost "**true**" ako i samo ako objekat nad kojim je primijenjena predstavlja nepraznu kolekciju, u suprotnom (tj. ako je kolekcija prazna) daje kao rezultat logičku vrijednost "**false**". Klasa "Narudzbe" posjeduje i preklopljeni operator "[]" koji kao rezultat daje ukupnu cijenu narudžbi koje je naručio student čije se ime navodi unutar uglastih zagrada (tj. zbir cijena svih narudžbi tog studenta). Ukoliko student sa navedenim imenom nije ništa naručio, ovaj operator kao rezultat daje nulu. Posljednja metoda ove klase je metoda "UcitajIzDatoteka" koja učitava podatke o narudžbama na osnovu ulaznih podataka koji su pohranjeni u dvije tekstualne datoteke čija se imena zadaju kao parametri (eventualno postojeće informacije o narudžbama se ne brišu, nego se samo vrši dopunjavanje kolekcije narudžbi novim narudžbama). Prva datoteka predstavlja datoteku sa podacima o tome

ko je šta naručio. Ova datoteka organizirana je tako da se za svaku narudžbu u prvom redu nalazi ime naručioca, u drugom redu naziv obroka, a u trećem redu naziv pića (treći red se ostavlja prazan ukoliko naručioc nije naručio piće). Narudžbe su u datoteci pohranjene u redoslijedu pristizanja. Na primjer, ova datoteka bi mogla izgledati recimo ovako:

```
Šaban Šaulić
Ćevapi u kajmaku
Jogurt
Zdravko Čolić
Krompiruša

Rambo Sulejmanović
Begova čorba
Gusti sok
Zdravko Čolić
Čizburger
Coca Cola "Vječiti student"
```

Druga datoteka predstavlja datoteku cijena. Ova datoteka organizirana je tako da se za svaki obrok ili piće u prvom redu nalazi naziv obroka ili pića, dok se u drugom redu nalazi cijena za navedeni obrok ili piće (u KM). Recimo, ova datoteka može izgledati recimo ovako:

```
Burek
3
Sirnica
2.5
Krompiruša
2
Ćevapi
5
Ćevapi u kajmaku
6
Begova čorba
3
Jogurt
1
Coca Cola "Vječiti student"
2.5
Gusti sok
2.5
```

Metoda "`UcitajIzDatoteka`" treba na osnovu sadržaja ovih datoteka da kreira odgovarajuće narudžbe i doda ih u kolekciju. U slučaju problema prilikom čitanja potrebno je baciti izuzetak tipa "`logic_error`" uz prateći tekst koji objašnjava prirodu problema. Taj tekst treba biti "Trazena datoteka ne postoji" u slučaju da jedna od tražene dvije datoteke ne postoji, "Nema odgovarajuce cijene" u slučaju da se u cjenovniku (datoteci cijena) ne nalaze informacije o traženom jelu ili piću, te "Problemi pri citanju" u ostalim slučajevima (npr. ukoliko datoteka sadrži besmislene podatke, poput nenumeričkih podataka tamo gdje bi se trebali nalaziti brojevi, ili u slučaju fizičkog oštećenja datoteke).

Napisane klase testirajte u programu koji će testirati sve njihove elemente.

4. Neka je *vozilo* objekat koji je, između ostalog, karakteriziran svojom težinom. *Automobil* je specijalna vrsta vozila, koje može primiti određeni manji broj putnika od kojih svaki ima svoju težinu. *Kamion* je specijalna vrsta vozila koje se može nakrcati teretom određene težine. *Autobus* je sličan automobilu, ali je predviđen za veći broj putnika.

   Cilj ovog zadatka je da razvijete surogatsku klasu "`Vozilo`" koja predstavlja polimorfni omotač za proizvoljnu vrstu vozila. Međutim, za tu svrhu, prvo je potrebno razviti hijerarhiju klasa koje opisuju pojedine vrste vozila. Bazna klasa "`ApstraktnoVozilo`" predstavlja apstraktnu baznu klasu koja sadrži ono što je zajedničko za sve razmatrane vrste vozila. Ona posjeduje konstruktor kojim se zadaje težina vozila (tipa cijeli broj), te metode nazvane "`DajTezinu`", "`DajUkupnuTezinu`", "`DajKopiju`" i "`IspisiPodatke`". Prva metoda daje vlastitu težinu vozila.. Druga metoda je u baznoj klasi apstraktna i predviđena je da daje ukupnu težinu vozila u koju je uračunata i težina svega što se u vozilu nalazi. Treća metoda je također apstraktna, a predviđena je da kreira kopiju objekta nad kojim je pozvana i da vrati kao rezultat adresu kreirane kopije. Ova metoda će služiti

za potrebe polimorfnog kopiranja. Konačno, apstraktna je i metoda "`IspisiPodatke`", koja je namijenjena za ispis podataka o tipu vozila, te njegovoj vlastitoj i ukupnoj težini. Naravno, sve apstraktne metode moraju biti izvedene tako da ukoliko se svim opisanim objektima pristupa preko pokazivača ili reference na baznu klasu "`ApstraktnoVozilo`", uvijek treba da bude pozvana ispravna verzija metode, koja će uzeti u obzir specifičnosti objekta.

Klasa "`Automobil`" nasljeđuje se iz klase "`ApstraktnoVozilo`". Njen konstruktor ima dodatni parametar, koji predstavlja vektor težina putnika (sve težine su cijeli brojevi). Zbog činjenice da postoji automatska konverzija inicijalizacionih listi u vektore, biće moguće konstrukcije tipa "`Automobil a(700, {80, 90, 60})`". Također, ova klasa sadrži konkretne realizacije metoda "`DajUkupnuTezinu`", "`DajKopiju`" i "`IspisiPodatke`". Ispis treba da izgleda poput sljedećeg:

```
Vrsta vozila: Automobil
Vlastita tezina: 700 kg
Tezine putnika: 80 kg, 90 kg, 60 kg
Ukupna tezina: 930 kg
```

Klasa "`Kamion`" se također nasljeđuje iz klase "`ApstraktnoVozilo`", a njen konstruktor posjeduje dodatni cjelobrojni parametar koji predstavlja težinu tereta. Naravno, ova klasa sadrži konkretne realizacije svih apstraktnih metoda. Ispis za ovu klasu treba da izgleda poput sljedećeg:

```
Vrsta vozila: Kamion
Vlastita tezina: 1600 kg
Tezina tereta: 850 kg
Ukupna tezina: 2450 kg
```

Klasa "`Autobus`" prilično je slična klasi "`Automobil`", tako da se i ona također nasljeđuje iz klase "`ApstraktnoVozilo`". Kako autobus može prevoziti mnogo putnika, ne čuva se informacija o težini svakog od njih, nego se čuva informacija o broju putnika i njihovoj prosječnoj težini (tako da je ukupna težina svih putnika jednaka proizvodu broja putnika i prosječne težine). Konstruktor ove klase ima dva dodatna parametra u odnosu na baznu klasu (broj putnika i prosječna težina jednog putnika), te konkretne realizacije svih apstraktnih metoda. Ispis za ovu klasu treba da izgleda poput sljedećeg:

```
Vrsta vozila: Autobus
Vlastita tezina: 2500 kg
Broj putnika: 30
Prosjecna tezina putnika: 75 kg
Ukupna tezina: 4750 kg
```

Promjenljive tipa "`Vozilo`" moraju biti takve da se u njih može smjestiti bilo automobil, bilo kamion, bilo autobus (tj. sadržaj promjenljive tipa "`Automobil`", "`Kamion`" ili "`Autobus`") odnosno promjenljiva bilo kojeg tipa koji je izveden iz apstraktnog tipa "`ApstraktnoVozilo`" (što uključuje i tipove koji će eventualno biti kreirani u budućnosti). Naravno, sa promjenljivim tipa "`Vozilo`" mogu se raditi sve operacije koje se mogu raditi sa bilo kojom vrstom vozila (takvih operacija ovdje nema mnogo, ali to je samo da zadatak ne bude dugačak), mogu se bezbjedno kopirati, međusobno dodjeljivati, itd.

Napisane klase iskoristite u testnom program koji čita podatke o vozilima iz tekstualne datoteke "`VOZILA.TXT`" u vektor čiji su elementi vozila (tj. koji su tipa "`Vozilo`"), a zatim sortira vozila po ukupnoj težini u rastući poredak (koristeći funkciju "`sort`") i na kraju ispisuje ukupne težine vozila nakon sortiranja (svaka težina u posebnom redu). Svaki red datoteke sadrži podatke o jednom vozilu. Za slučaj automobila, prvi znak u redu je "`A`", nakon čega slijedi vlastita težina automobila, broj putnika i težina svakog od putnika, na primjer "`A500 3 80 60 75`" za automobil težine 500 kg sa 3 putnika težina 80, 60 i 75 kg respektivno. Za slučaj kamiona, prvi znak u redu je "`K`", nakon čega slijedi težina kamiona i težina tereta, na primjer "`K1500 1200`" za kamion težine 1500 kg natovaren sa teretom težine 1200 kg. Konačno, za slučaj autobusa, prvi znak u redu je "`B`", nakon čega slijedi težina autobusa, broj putnika i prosječna težina putnika, na primjer "`B2200 50 80`" za autobus težine 2200 kg sa 50 putnika čija je prosječna težina 80 kg. U slučaju bilo kakvih problema pri čitanju datoteke (što uključuje i slučaj kada datoteka sadrži neispravne podatke) treba ispisati prikladne poruke o greški.

5. Dopunite generičku klasu "Matrica" koju ste razvili u Zadatku 6 sa Tutorijala 12 (a koja se oslanja na klasu razvijenu na Predavanju 11_b) sa četiri nove metode "SacuvajUTekstualnuDatoteku", "SacuvajUBinarnuDatoteku", "ObnoviIzTekstualneDatoteke" i "ObnoviIzBinarneDatoteke", te jednim dodatnim konstruktorom, koji će kasnije biti opisan (ovo je ujedno dobra prilika da dovršite Zadatak 6 sa Tutorijala 12 ukoliko to već niste uradili prije). Sve ove metode primaju kao parametar naziv datoteke. Metoda "SacuvajUTekstualnuDatoteku" snima sadržaj matrice nad kojom je pozvana u tekstualnu datoteku čije je ime zadano. Datoteka treba formatirana tako da se podaci o svakom redu matrice čuvaju u posebnim redovima datoteke, pri čemu su elementi unutar jednog reda međusobno razdvojeni zarezima (iza posljednjeg reda nema zareza). Recimo, za neku matricu formata 3×4 kreirana datoteka može izgledati poput sljedeće:

   ```
   2.5,-3,1.12,4
   0,0.25,3.16,42.3
   -1.7,2.5,0,5
   ```

   U slučaju da dođe do bilo kakvih problema pri upisu, treba baciti izuzetak tipa "logic_error" uz prateći tekst "Problemi sa upisom u datoteku". Metoda "SacuvajUBinarnuDatoteku" obavlja sličnu funkcionalnost, samo što se upis vrši u binarnu datoteku, u koju je potrebno snimiti one podatke iz memorije koji su neophodni da bi se kasnije mogla pouzdano izvršiti rekonstrukcija stanja matrice iz pohranjenih podataka.Pri tome se podrazumijeva da je tip elemenata matrice neki skoro-POD tip podataka (u suprotnom, snimanje u binarnu datoteku najvjerovatnije neće biti uspješno). U slučaju problema pri upisu, vrijedi isto kao kod prethodne metode. Dalje, metode "ObnoviIzTekstualneDatoteke" odnosno "ObnoviIzBinarneDatoteke" vrše obnavljanje sadržaja matrice na osnovu sačuvanog stanja u tekstualnoj odnosno binarnoj datoteci, pri čemu se prethodni sadržaj matrice uništava. U oba slučaja, ukoliko tražena datoteka ne postoji, treba baciti izuzetak tipa "logic_error" uz prateći tekst "Trazena datoteka ne postoji". Pri čitanju iz tekstualne datoteke, ukoliko datoteka sadrži podatke koji nisu u skladu sa tipom elemenata matrice, ukoliko podaci nisu razdvojeni zarezima, ili ukoliko različiti redovi imaju različit broj elementa, treba baciti isti izuzetak, uz prateći tekst "Datoteka sadrzi besmislene podatke" (s obzirom da u tekstualnoj datoteci nije pohranjena nikakva informacija o broju redova i kolona, datoteku ćete morati efektivno iščitati dva puta, prvi put da saznate broj redova i kolona, a drugi put da zaista pročitate vrijednosti elemenata, nakon što su obavljene odgovarajuće dinamičke alokacije). U slučaju bilo kakvih drugih problema pri čitanju, treba također baciti isti izuzetak, uz prateći tekst "Problemi pri citanju datoteke". Za slučaj čitanja iz binarne datoteke, u slučaju bilo kakvih problema (osim nepostojeće datoteke), treba baciti ovaj isti izuzetak.

   Konačno, rečeno je da je u klasu "Matrica" potrebno dodati i novi konstruktor. Taj konstruktor će imati dva parametra, pri čemu je prvi parametar ime datoteke, a drugi je logička vrijednost koja određuje da li će se konstrukcija objekta vršiti iz tekstualne datoteke (u slučaju kad taj parametar ima vrijednost "**false**") ili iz binarne datoteke (kada parametar ima vrijednost "**true**"). Ovaj konstruktor ponaša se identično kao što se ponašaju metode "ObnoviIzTekstualneDatoteke" odnosno "ObnoviIzBinarneDatoteke", samo se oslanja na činjenicu da se objekat tek stvara, tako da nema potrebe za oslobađanjem resursa koje je objekat prije toga koristio.

   Obavezno napišite i kratki testni program u kojem ćete testirati novododane funkcionalnosti klase vezane za datoteke. Ostale funkcionalnosti klase ne morate testirati, s obzirom da se podrazumijeva da ste ih testirali ranije. Naravno, dozvoljena je upotreba i ostalih funkcionalnosti ukoliko su Vam potrebne za potrebe testiranje onoga što se traži da testirate (recimo, da formirate matrice koje želite snimiti, ili da ispišete sadržaj obnovljene matrice).

6. Potrebno je napraviti generičku funkciju "SortirajBinarnuDatoteku" koja omogućava sortiranje podataka pohranjenih u binarnoj datoteci *bez učitavanja sadržaja datoteke u memoriju*. Funkcija treba da ima sljedeći prototip:

   ```
   template <typename TipElemenata>
     void SortirajBinarnuDatoteku(const char ime_datoteke[],
       std::function<bool(TipElemenata, TipElemenata)> kriterij
         = std::less<TipElemenata>());
   ```

   Prvi parametar je ime datoteke, za koju se podrazumijeva da je organizirana kao kolekcija elemenata istog tipa "TipElemenata" (koji naravno mora biti neki skoro-POD tip), snimljenih u

datoteku jedan za drugim. Pri tome je posve nebitno kako je ta kolekcija nastala (da li snimanjem elemenata jedan za drugim, "istresanjem" čitavog niza elemenata odjednom, ili na neki treći način). Drugi parametar je funkcija kriterija, koja radi na posve analogan način kao funkcija kriterija u bibliotečkoj funkciji "`sort`", pri čemu se može zadati bilo obična funkcija, bilo lambda funkcija, bilo funkcijski objekat (funktor), tačnije bilo šta što se može pozvati sa dva argumenta koji su tipa "`TipElemenata`" (ili se mogu konvertovati u taj tip) i koja vraća logičku vrijednost (ili nešto što se može interpretirati kao logička vrijednost). Recimo, ukoliko datoteka "`BROJEVI.DAT`" sadrži cijele brojeve, njen sadržaj možemo sortirati u opadajući poredak pozivom poput

```
SortirajBinarnuDatoteku<int>("BROJEVI.DAT", [](int x, int y) { return x > y; });
```

Nažalost, tip elemenata se mora eksplicitno specificirati (tj. nije moguće pisati samo nešto poput "`SortirajBinarnuDatoteku(...)`" umjesto "`SortirajBinarnuDatoteku<int>(...)`"), odnosno tip se ne može deducirati iz parametara lambda funkcije, jer je dedukcija moguća samo u slučaju potpunog slaganja tipa (drugi parametar je deklariran kao polimorfni funkcijski omotač, kojem se zaista može dodijeliti lambda funkcija, ali lambda funkcija nije polimorfni funkcijski omotač, pa nemamo potpuno slaganje po tipu). Koristeći funkcijske objekte iz bibiloteke "`functional`", prethodni poziv mogli smo kraće izvesti kao

```
SortirajBinarnuDatoteku<int>("BROJEVI.DAT", std::greater<int>());
```

Drugi parametar ima podrazumijevanu vrijednost "`std::less<TipElemenata>()`", tako da se može izostaviti ukoliko želimo sortiranje u podrazumijevani rastući poredak.

U slučaju da zadana datoteka ne postoji, treba baciti izuzetak tipa "`logic_error`" uz prateći tekst "Datoteka ne postoji". U slučaju bilo kakvih drugih problema pri čitanju ili pisanju datoteke treba baciti isti izuzetak uz prateći tekst "Problemi u pristupu datoteci".

Napisanu funkciju iskoristite u testnom programu koji će testirati rad ove funkcije. Taj testni program moraće prvo kreirati neku binarnu datoteku, zatim pozvati ovu funkciju, i onda konačno iščitati njen sadržaj, da se uvjerimo da je datoteka zaista sortirana. Najstrože je zabranjeno da funkcija "`SortirajBinarnuDatoteku`" učita sadržaj datoteke u memoriju, sortira ga u memoriji i zatim vrati sortirani sadržaj nazad. Sortiranje se mora obaviti isključivo manipuliranjem nad sadržajem datoteke. Kršenje ove zabrane rezultiraće *dodjelom nula poena na ovaj zadatak*. Za sortiranje koristite neki od naivnih algoritama sortiranja koji Vam je poznat. Vodite računa da, uz pogodne funkcije kriterija, ova funkcija mora biti u stanju da sortira obje datoteke "`BROJEVI.DAT`" i "`STUDENTI.DAT`" koje su demonstrirane na Predavanju 14_b!